Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051675

ARTICLE

A Graph with Adaptive Adjacency Matrix for Relation Extraction

Run Yang[1,2,3], Yanping Chen[1,2,3,*], Jiaxin Yan[1,2,3] and Yongbin Qin[1,2,3]

[1]Engineering Research Center of Text Computing & Cognitive Intelligence, Ministry of Education, Guizhou University, Guiyang, 550025, China

[2]State Key Laboratory of Public Big Data, Guizhou University, Guiyang, 550025, China

[3]College of Computer Science and Technology, Guizhou University, Guiyang, 550025, China

*Corresponding Author: Yanping Chen. Email: ypench@gmail.com

Received: 12 March 2024 Accepted: 25 July 2024 Published: 12 September 2024

ABSTRACT

The relation is a semantic expression relevant to two named entities in a sentence. Since a sentence usually contains several named entities, it is essential to learn a structured sentence representation that encodes dependency information specific to the two named entities. In related work, graph convolutional neural networks are widely adopted to learn semantic dependencies, where a dependency tree initializes the adjacency matrix. However, this approach has two main issues. First, parsing a sentence heavily relies on external toolkits, which can be error-prone. Second, the dependency tree only encodes the syntactical structure of a sentence, which may not align with the relational semantic expression. In this paper, we propose an automatic graph learning method to autonomously learn a sentence's structural information. Instead of using a fixed adjacency matrix initialized by a dependency tree, we introduce an Adaptive Adjacency Matrix to encode the semantic dependency between tokens. The elements of this matrix are dynamically learned during the training process and optimized by task-relevant learning objectives, enabling the construction of task-relevant semantic dependencies within a sentence. Our model demonstrates superior performance on the TACRED and SemEval 2010 datasets, surpassing previous works by 1.3% and 0.8%, respectively. These experimental results show that our model excels in the relation extraction task, outperforming prior models.

KEYWORDS

Relation extraction; graph convolutional neural network; adaptive adjacency matrix

1 Introduction

Relation extraction (RE) is a critical sub-task of information extraction that aims to identify the semantic relationship between two named entities in a sentence. As a fundamental task, it has been widely adopted to support many downstream applications, such as knowledge graphs [1,2], biomedical knowledge discovery [3,4], and question answering [5,6]. Despite significant achievements in this field, extracting relations from sentences remains a challenging task because sentences often contain multiple entities, and each entity pair must be verified. Since all entity pairs share the same contextual features

within a sentence, it is essential to learn a structured sentence representation that encodes dependency information specific to the two named entities.

Many strategies have been proposed to learn the structural information of a sentence for the relation extraction task, e.g., position embedding [7], multi-channel [8], graph convolutional network [9–11] and neuralized feature engineering [12,13]. Among these, the graph convolutional network (GCN) stands out as the most popular strategy, underpinned by a solid theoretical foundation in computational linguistics, particularly dependency grammar. In a GCN, token representations serve as the nodes of the graph, with the edges between nodes represented by a matrix typically initialized by a dependency tree. The GCN excels at learning abstract sentence representations that encode structural information pertinent to a dependency tree. Compared to other dependency tree-based models, such as Long Short Term Memory (LSTM) networks [14,15], the GCN can directly take a tree structure as input, making it more suitable for representing the semantic structure of a sentence. Consequently, in recent years, GCN have been extensively studied to support relation extraction.

In dependency tree-based GCN models, external toolkits are required to generate the dependency tree for initializing the adjacency matrix. To mitigate the influence of errors in a dependency tree, manually designed strategies are often employed to prune the tree. In relation extraction tasks, researchers commonly utilize dependency trees parsed by external tools as adjacency matrices for GCNs. For example, Zhang et al. [10] proposed a "hard pruning strategy," using the lowest common ancestor (LCA) of two entities in a dependency tree to convert it into a graph. Guo et al. [16] designed an attention-guided "soft pruning" strategy to transform the dependency tree into a fully connected weighted graph. Zhou et al. [17] used the dependency tree to construct a logical adjacency matrix (LAM) that captures k-order neighborhood information. Tian et al. [18] presented a dependency-driven attention graph convolutional network to differentiate the importance of various word dependencies in the dependency tree. Zhang et al. [19] developed a multi-scale graph convolutional network to obtain both rough and refined views of dependency trees.

Because generating the dependency tree requires parsing a whole sentence, the task is even more challenging than relation extraction itself and remains error-prone. Furthermore, dependency trees only encode the syntactical information of a sentence, which may not align with the semantic expression of relations. To avoid reliance on dependency trees, some researchers have shifted to constructing graph structures for GCNs using entities directly as nodes or by utilizing feature combinations. For example, Vashishth et al. [20] treated entities as nodes and established connections between them, thereby constructing a multi-relational graph capable of recognizing various types of entity relationships. Xu et al. [21] utilized prior knowledge to combine atomic features to construct graph structures for relation extraction. Although these methods avoid dependence on external tools, limiting the graph nodes to key entities restricts the model's ability to capture more granular features. Additionally, feature combinations rely on predefined rules, and the adjacency matrix of the graph remains fixed, making it difficult for the model to flexibly learn structural information.

To address these limitations, we propose a Graph Automatic Learning Deep Network (GALDN) for relation extraction. Instead of using entities as nodes, we employ more fine-grained token representations as graph nodes. We no longer initialize the adjacency matrix using dependency trees or prior knowledge; instead, we initialize it by computing the correlation of token representations and dynamically updating the adjacency matrix to better encode the structural information of sentences. Specifically, given a sentence, a pre-trained language model transforms each token into an abstract representation encoding its contextual features and semantic information. Subsequently, semantic dependencies between tokens are learned from their correlations, where higher correlations indicate

stronger semantic dependencies. In our model, an adaptive adjacency matrix is initialized by computing correlations between token representations. During training, the adjacency matrix is updated based on token representations, enabling our network to flexibly encode semantic dependencies and structural information. We evaluate our method on the TACRED and SemEval 2010 datasets, achieving F1 scores of 71.8% and 91.1%, respectively. Compared to baseline models, our performance improves by 1.0% and 0.8%, respectively. Analytical experiments demonstrate GALDN's ability to construct adjacency matrices for each input, effectively learning the semantic structure of sentences. The contributions of this work are summarized as follows:

We propose a new Graph Automatic Learning Deep Network that can learn the semantic structure of sentences without relying on external toolkits or prior knowledge, with the capability to learn structural information.

Extensive experiments on the TACRED and SemEval 2010 datasets demonstrate the effectiveness of our model.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 describes the construction process of our model. Section 4 provides details on the experiments. Section 5 presents our conclusions and future work.

2 Related Work

In relation extraction, it is essential to learn a structured sentence representation that encodes dependency information relevant to two specific named entities in a sentence. Various strategies have been proposed to achieve this. In this section, we categorize them into three main approaches: feature-based methods, language model-based methods, and graph-based methods.

2.1 Feature-Based Methods

Feature-based methods capture the relationships between entities through carefully designed features, typically including part-of-speech tagging, entity types, dependency paths, and other syntactic information. For example, Kambhatla [22] studied the impact of various features such as entity vocabulary and entity types on the performance of relation extraction. Zhou et al. [23] proposed using basic phrase structures as shallow syntactic features combined with SVM models to improve the accuracy of relation extraction. Chen et al. [12] captured semantic and structural information of relation instances by combining atomic features. Li et al. [24] extracted the overall semantic features of sentences and introduced syntactic dependency features into the model for relation extraction.

Feature-based methods generally involve well-defined features, making them interpretable. However, these methods heavily rely on prior knowledge, requiring researchers to have a high level of expertise. This approach may overfit evaluation datasets and transferring them across different domains is complex.

2.2 Language Model-Based Methods

In recent years, pre-trained language models, especially large language models, have been widely applied in relation extraction due to their outstanding data processing and language understanding capabilities. Models such as GPT [25], BERT [26] and its variants (SpanBERT [27]), as well as LlaMa [28], LasUIE [29], and other large language models based on the Transformer framework, are pretrained on large corpora to capture deep semantic information.

Researchers often fine-tune these language models to adapt them to relation extraction tasks. For instance, Li et al. [30] proposed a method for sentence bi-dimensional representation using BERT as the model encoder. Cho et al. [31] converted dependency tree features into positional encodings and combined them with the BERT pre-trained language model. Veyseh et al. [32] utilized BERT as an encoding module, extracting syntactic importance scores of words based on dependency trees to introduce syntactic information into the model for better generalization. Liang et al. [33] employed CNN and attention mechanisms to obtain semantic features at different granularities. Subsequently, pre-trained language models like BERT and SpanBERT are fine-tuned and applied to support relation extraction tasks. Wei et al. [34] used entity type representations as entity annotations, inserting them near the positions of entities in the sentence to aggregate more semantic information from pre-trained language models for masked token prediction in prompt-tuning.

Fine-tuning language models for relation extraction optimizes them to better understand task-relevant semantic information, thereby improving performance and helping to reduce the risk of overfitting. However, the underlying networks of existing language models utilize serialized sentence representations, typically representing linear semantic relationships within sentences, which poses challenges in parsing complex semantic structures.

2.3 Graph-Based Methods

Due to the effectiveness of graph neural networks in learning graph structures, many researchers use these networks to capture the semantic structure of sentences. Sentence-level dependency trees are often utilized to construct the graph structure, enabling the graph neural network to extract the semantic information from the sentence. Many researchers input the entire dependency tree into a graph convolutional neural network for relation extraction [35–39]. For example, Zhang et al. [10] used the nearest common ancestor subtree of two entities in the dependency tree to construct a graph structure for relation extraction. Miwa et al. [14] considered not only the nearest common ancestor subtree and the shortest path subtree between two entities in the dependency tree but also the entire dependency tree, feeding them into the graph convolutional neural network for comparison. Fei et al. [35] explicitly integrated dependency and discourse structures to provide richer semantic and feature learning for relation extraction.

Due to parsing errors, some researchers have moved away from constructing graph structures with dependency trees and instead directly construct graph structures based on the entities themselves. For example, Vashishth et al. [20] treated entities as nodes and constructed edges between them, creating a multi-relational graph to identify various relationship categories between entities. Sun et al. [40] used entities and entity pairs to construct bipartite graphs, where nodes are entities and entity pairs, and edges exist only between entities and entity pairs. Zhu et al. [41] designed a graph neural network model that generates parameters by treating entities as nodes, improving the model's ability to reason in multi-hop relations.

However, the aforementioned research has the following issues: In GCN models based on dependency trees, the adjacency matrix is typically initialized using dependency trees parsed by external tools, which may introduce errors and unnecessary noise. In GCN models where entities serve as graph nodes, limiting the graph nodes to only key entities may restrict the model from capturing finer-grained features and may fail to fully utilize contextual information surrounding the entities.

3 Model

In this section, we first introduce the task definition of relation extraction, followed by a description of the structure of our model, which is composed of four modules as shown in Fig. 1: Encoding Module, Graph Automatic Learning Deep Network, Entity Attention Residual Module, and Relationship Classification Module. Each module is discussed in detail below.

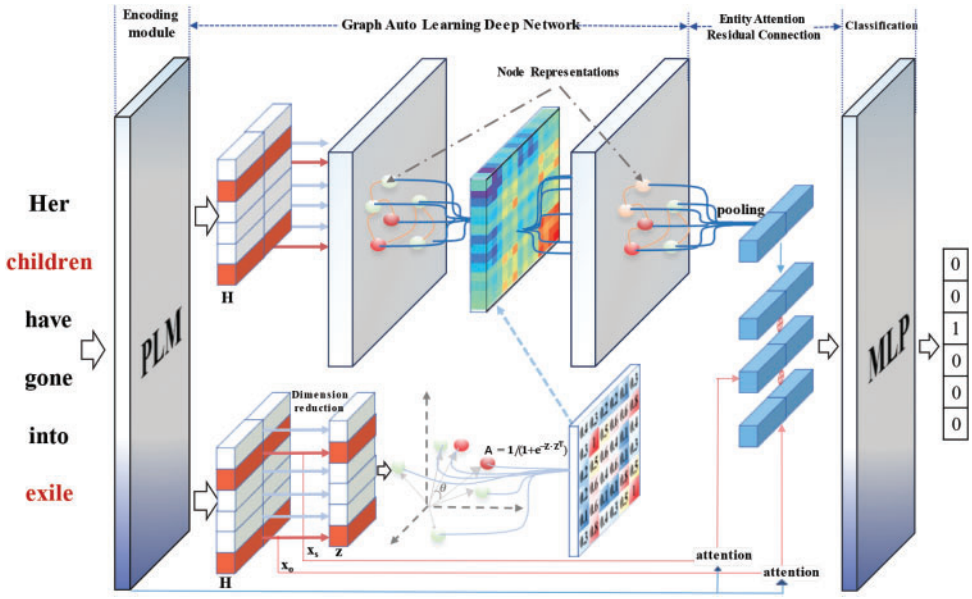

Figure 1: The above figure shows the overall architecture of the model, which is mainly composed of four parts: Ecoding module, graph automatic learning deep network, entity attention residual and classification module

3.1 Definition of Relation Extraction Task

Given a sentence $\mathbf{S} = [x_1, x_2, ..., x_N]$ composed of N tokens, let $e_1 = [x_i, ..., x_j]$ and $e_2 = [x_s, ..., x_t]$ be two named entities in $\mathbf{S}$. The task of relation extraction is to assign a relation label y_i, where $y_i \in \mathbf{Y} = [y_0, ..., y_K]$ for the entity pair (e_1, e_2), where K is the number of predefined relation types and y_0 indicates that there is no predefined relation types between e_1 and e_2.

$$\mathcal{F}: (\mathbf{S}) \rightarrow \mathbf{Y} \tag{1}$$

In the formula, $\mathcal{F}$ represents a classifier (GCN, NN, etc.) used to map the set of relation instances to the corresponding set of labels.

3.2 Encoding Module

Before feeding into the graph neural network, the sentence $\mathbf{S}$ is transformed into an abstract representation by a pre-trained language model (PLM). The process is represented as:

$$\mathbf{H} = \text{PLM}([x_1, x_2, \ldots, x_N]) \tag{2}$$

where, $\mathbf{H} = [h_1, h_2, ..., h_N]$, which h_i is an abstract representation of token x_i. Because the PLM is trained on external resources using an unsupervised method, it effectively learns the semantic information and dependencies of token x_i in sentence $\mathbf{S}$.

In our experiments, we adopt SpanBERT, proposed by Joshi et al. [27], to implement the encoding module. In SpanBERT, every token is transformed into a 1024-dimensional vector. These token representations are used as nodes in the graph neural network discussed below.

3.3 Graph Automatic Learning Deep Network

A Deep Graph Convolutional Network (GCN) is formalized as an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with n words, where $\mathbf{V} = \{v_1, v_2, ..., v_N\}$ REthe node set and $\mathbf{E} = \{(v_i, v_j)| v_i \in \mathbf{V}, v_j \in \mathbf{V}\}$ represent the node and edge set, respectively. An edge $(v_i, v_j) \in \mathbf{E}$ indicates indicates a connection between node v_i and v_j. The edges can also be denoted as a matrix $\mathbf{A}$ known as "adjacency matrix".

In a traditional model, the node set is composed of abstract token representations that $\mathbf{V} = \{v_i = h_i | h_i \in \mathbf{H}\}$. The adjacency matrix $\mathbf{A}$ is initialized by a optimized dependency tree, which is generated by a parsing toolkit. Given a token representation set $\mathbf{H}$ and an initialized adjacency matrix $\mathbf{A}$, the output of l-th layer of a GCN is formalized as:

$$h_i^{(l)} = \sigma\left(\sum_{j=1}^{N} \frac{\tilde{A}_{ij} h_j^{(l-1)} W^l}{d_i} + b^{(l)}\right) \tag{3}$$

Here, l represents the number of layers in the GCN, and its value ranges from 0 to L. $\tilde{A} = A + I$, I is the identity matrix. Elements of $\tilde{A}$ are denoted as $\tilde{A}_{ij}$, $d_i = \sum_{j=1}^{N} \tilde{A}_{ij}$ is the degree of the i-th node, $h^{(0)} = \mathbf{H}$, which is the output of the employed PLM. W^l is a linear transform matrix, $b^{(l)}$ is the bias term.

As previously discussed, in traditional models, the adjacency matrix $\mathbf{A}$ is initialized by a dependency tree, but parsing the dependency tree with external tools is prone to errors, and the syntactic structure may not correspond to the semantic expression of a relation. Therefore, the initialization of the adjacency matrix in our model is not achieved through prior knowledge or external toolkits but by multiplying token representations, calculating the association between tokens in a sentence. Specifically, our model treats each token representation in the sentence as a node of the graph, and then initializes the adjacency matrix $\mathbf{A}$ by multiplying token representations. The larger the value obtained from the multiplication of two token representations, the smaller the angle between them in the feature space, indicating that the two tokens are more related, and the semantic dependency between the tokens is stronger. The corresponding weight in the adjacency matrix will be larger. This adjacency matrix initialized by such correlation can reflect the semantic structural information between tokens in a sentence. The process is formalized as follows:

$$\mathbf{A} = \frac{1}{\left(1 + e^{-\mathbf{Z} \times \mathbf{Z}^{\mathrm{T}}}\right)} \tag{4}$$

where, T is the matrix transpose operation. $\mathbf{Z} = \varphi(\mathbf{H})$ is a dimension reduction function. We use a linear layer as the dimension reduction function. It transforms the token representation $\mathbf{H} = [h_1, h_2, ..., h_N]$ into a lower representation $\mathbf{Z} = [z_1, z_2, ..., z_N]$, where $z_i = \varphi(h_i)$. In our experiment, the h_i is the output of the SpanBERT model. It is a 1024-dimensional vector. The reduced representation z_i is a 16-dimensional vector. Elements of $\mathbf{A}$ are denoted as $\mathbf{A}_{\mathrm{ij}}$. They take values between 0 and 1, which indicate the intensity of semantic dependency between token x_i and x_j.

After initializing the adjacency matrix **A**, it is involved in model training. During the training process, the adjacency matrix **A** is updated according to the token representations. If the semantic association between two tokens is stronger, the weight at the corresponding position in the adjacency matrix **A** will increase; otherwise, it will decrease. This optimizes the values of the elements in the adjacency matrix **A**, enabling the token representations to encode the semantic dependency of a sentence. We refer to this adjacency matrix **A** as the Adaptive Adjacency Matrix (AAM).

3.4 Entity Attention Residual Connection

This module is designed to mitigate the vanishing gradient problem when multiple GCN layers are stacked. We use max pooling to obtain the entity representation. The entity representation is then used as a query to extract semantic information at the entity level from **H**. The formulas are as follows:

$$x_s = \underset{\text{e1}}{\text{MaxPooling}}(\mathbf{H}) \tag{5}$$

$$x_o = \underset{\text{e2}}{\text{MaxPooling}}(\mathbf{H}) \tag{6}$$

$$x_s^{'} = \text{Softmax}(\mathbf{H}\text{x}_\text{s}/\sqrt{d})\mathbf{H} \tag{7}$$

$$x_o^{'} = \text{Softmax}(\mathbf{H}\text{x}_\text{o}/\sqrt{d})\mathbf{H} \tag{8}$$

where, **H** is the output of the SpanBERT. x_s and x_o are the representations of the subject (e_1) and object (e_2) obtained through the MaxPooling function. d is the dimension of the vector representation, used to normalize the token representations. The semantic information representations $x_s^{'}$ and $x_o^{'}$ are obtained through the attention mechanism, which contains more entity attention.

The entity attention residual module encodes important semantic information previously attended to by the old entity representations through cross-layer connections. This process is shown as follows:

$$h_{sent} = F\left(\mathbf{h}^{(\mathbf{L})}\right) = F\left(\text{GCN}\left(\mathbf{h}^{(\mathbf{0})}\right)\right) \tag{9}$$

$$h_{out} = [h_{sent} \oplus x_s^{'} \oplus x_o^{'}] \tag{10}$$

where, $\mathbf{h}^{(\mathbf{L})}$ is the output of layer L of the GCN. F is a max pooling function, h_{sent} is the result of applying max pooling to $\mathbf{h}^{(\mathbf{L})}$, $\oplus$ denoting the vector concatenation operation. h_{out} is the final representation obtained by concatenating h_{sent} with the output of the entity attention residual module.

3.5 Relation Classification Module

In this module, h_{out} is fed into a Multi-Layer Perception (MLP) to obtain the final fused representation h_{final}. It is formalized as follows:

$$h_{final} = ReLU\left(W_m h_{out} + b\right) \tag{11}$$

where, W_m is a linear transformation matrix, ReLU is a nonlinear activation function, and b is the bias term. Next, h_{final} is further fed into a fully connected layer followed by a softmax function. The probability distribution for all relation types is computed as follows:

$$p\left(y \mid h_{final}\right) = Softmax\left(Wh_{final} + b\right) \tag{12}$$

Finally, the predicted result is the most likely relationship category. We use the minimum cross-entropy as the loss function. During training, the values of the adaptive adjacency matrix are optimized by task-relevant learning objectives, which dynamically learn the semantic dependency between tokens.

4 Experiment

4.1 Datasets

In the experiments, our model is evaluated on two datasets: TACRED and SemEval 2010. TACRED (TAC Relation Extraction Dataset) is a large dataset commonly used in relation extraction. It contains 106,264 instances from news articles and online texts in the corpus used by the TAC competition. It contains 41 relation types. To compare with related works, the training, validation, test sets contains 68,124, 22,631, 15,509 instances, respectively. For the TACRED dataset, we follows the implementation details mentioned by Zhang et al. [42]. We report the final performance by averaging results over five different seeds. SemEval 2010 is a public dataset that contains nine relation classes. The training and test sets contain 7000 and 2717 instances, respectively. The statistical information of datasets is shown in Table 1.

Table 1: Details of datasets

Datasets	Train	Validation	Test	RelaPatons
TACRED	68124	22631	15509	41
SemEval 2010	8000	/	2717	9

To evaluate the performance of our model on these two datasets, we use the traditional precision (P), recall (R), F1 score (F1). The following formula is calculated:

$$P = \frac{TP}{TP + FP} \tag{13}$$

$$R = \frac{TP}{TP + FN} \tag{14}$$

$$P = \frac{2 \times P \times R}{P + R} \tag{15}$$

where, TP is the number of true positive instances in the prediction results. A true positive instance means that the label is positive and the prediction is positive. FP is the number of false positive instances, meaning that the label is negative but the prediction is positive. FN is the number of false negative instances, meaning that the label is positive but the prediction is negative.

4.2 Setting

Following the implementation of Zhang et al. [42], we use a special [SUBJ-<NER>] marker to replace each subject entity and object entity in the original sentences in the TACRED dataset. In the SemEval 2010 dataset, we tag entities with <e1>, </e1>, <e2>, and </e2> at both ends. Our model is based on the PyTorch framework and was trained in an experimental environment using an A100 GPU. We used Adam as the optimizer, with the random seed set to 42. For the TACRED dataset, we set the learning rate to 3e-5, batch size to 64, number of GCN layers to 2, and maximum sentence length to 128. For the SemEval 2010 dataset, we set the learning rate to 2e-5, batch size to 32, number

of GCN layers to 3, and maximum sentence length to 128. We used SpanBERT-large as our encoder, which outputs 1024-dimensional vectors. To construct the adjacency matrix, the dimension of each token is reduced to 16-dimensional vectors. The main parameters used in the experiments are shown in Table 2.

Table 2: Parameter settings

	TACRED	SemEval 2010
Optimizer	Adam	Adam
lr	3e-5	2e-5
Warmup steps	300	0
Batch size	64	32
Seed	42	42
Epochs	5	10
GCN layer	2	3
Max length	128	128

4.3 Comparison with Related

In this section, we compare our model with related works on the TACRED dataset. Related works can be roughly divided into three categories: LSTM-based models, GCN-based models, and models based on pre-trained language models.

(1) LSTM-based models. PA-LSTM: Zhang et al. [42] introduced an attention mechanism of position in the LSTM network. SDP-LSTM: Xu et al. [43] proposed a SDP-LST Mmodel, which focuses more on relevantin formation and ignore the irrelevant.

(2) GCN-based models. C-GCN: Zhang et al. [10] proposed a hard pruning strategy to prune the dependency tree. AGGCN: Guo et al. [16] proposed an attention guided graph convolutional network with full dependency tree as input. EA-WGCN: Zhou et al. [17] constructed a logical matrix so that k-order domain information can be obtained by using a layer of GCN. These models are all constructedon the dependency tree. C-MDR-GCN: Hu et al. [44] utilized dependency trees in three different ways, combining the three types of adjacency matrices obtained to construct a graph structure. DAGCN: Zhang et al. [45] proposed a dual-attention graph convolutional network with a parallel structure, which can establish multi-round interactions between context and dependency information.C-DAGCN: Liao et al. [46] proposed a context-dependent aware graph convolutional network that perceives the importance of word dependency relationships from multiple perspectives across the entire dependency tree, without considering noise in the dependency tree.

(3) Based on pre-trained language models. Since we use SpanBERT as the model encoder,we also directly compare our model with the SpanBERT pre-trained model [27]. In the SemEval 2010 dataset, we also compared with the R-BERT model [47] and SpanBERT+SMS model [33]. R-BERT: The model sums and averages the word vectors contained in entities to obtain a vector representation of the entities, which is then concatenated with the sentence vector for relation extraction. SpanBERT+SMS: The model [33] proposed a multi-granularity feature extraction method for sentences, using SpanBERT as the encoder and employing CNN combined with an attention mechanism to capture multi-granularity features within sentences. The experimental results are shown in Tables 3 and 4.

Table 3: Performance on the TACRED dataset

Model	P (%)	R (%)	F1 (%)
SDP-LSTM [42]	66.3	52.7	58.7
PA-LSTM [43]	65.7	64.5	65.1
C-GCN [10]	69.9	63.3	66.4
AGGCN [16]	73.1	60.9	68.2
EA-WGCN [17]	70.8	64.8	67.6
C-MDR-GCN [44]	68.9	67.1	68.0
SpanBERT [27]	70.8	70.9	70.8
DAGCN [45]	72.4	64.8	68.4
Ours	73.7	70.0	71.8

Table 4: Performance on the SemEval 2010 dataset

Model	F1 (%)
SDP-LSTM [42]	83.7
PA-LSTM [43]	82.7
C-GCN [10]	84.8
AGGCN [16]	85.7
EA-WGCN [17]	85.4
R-BERT [47]	89.2
SpanBERT [27]	89.4
SpanBERT+SMS [33]	90.3
DAGCN [45]	86.0
C-DAGCN [46]	90.2
Ours	**91.1**

The experimental results on the TACRED dataset shows that LSTM-based models exhibit lower performance. The reason is that LSTM is a sequence model, which main learn linear seman-tic dependency in a sentence. On the other hand, expressions of relations have more complex seman-tic structures. In this condition, LSTM is not practical to capture the semantic dependency of a relation. Compared with LSTM based models, the GCN based models is implemented on dependency tress. They are more practical to learn the semantic structure of a relation. Therefore, they have better performance. However, traditional GCN models utilize dependency trees as the graph structure, the construction of which largely relies on external toolkits. Moreover, these external toolkits are prone to errors during the parsing process, introducing unnecessary noise and thus leading to decreased performance. Compared to other GCN models, our model no longer uses dependency trees parsed by external tools as the adjacency matrix. Instead, it initializes by calculating the association degree between tokens in a sentence, where the more related two tokens are, the stronger the semantic dependency between them, and the greater the weight at the corresponding position in the adaptive matrix. By involving the adaptive matrix in model training, it dynamically optimizes the elements

of the adaptive adjacency matrix, allowing the model to dynamically learn the semantic structure of sentences. Our model's F1 score is at least 6.7% higher than LSTM-based models, and at least 1.3% higher than GCN-based models. The F1 score is 1.0% higher than SpanBERT.

The performance on the SemEval 2010 dataset has the same trend as the TACRED dataset. Compared with related works, the F1 score of our model is at least 7.4% higher than that of the LSTM-based model and at least 5.7% higher than that of the GCN-based model. Our model outperforms the SpanBERT about 2.7% in F1 score. The F1 score of our model is 1.9% higher than R-BERT and 0.8% higher than SpanBERT+SMS.

In summary, even without depending on any external toolkit or prior knowledge, our model shows clear improvements on the two evaluation datasets, outperforming all LSTM and GCN-based models. The results demonstrate the effectiveness of our model in capturing the semantic structural information between tokens in a sentence.

4.4 Ablation Study

Our model is primarily composed of two modules: (1) The first module obtains the potential global semantic information of the sentence by auto matically learning and dynamically optimizing the Adaptive Adjacency Matrix. (2) The second module connects the semantic information focused at the entity level with the output of the adaptive module through cross-layer connections using a residual module, reducing the impact of the vanishing gradient problem when stacking multiple layers of GCN. To demonstrate the effectiveness of each module, this paper conducts ablation experiments on the SemEval 2010 and TACRED datasets. The experimental results on the two datasets are shown in Table 5, where "w/o" stands for "without" indicating that the model has removed the corresponding module.

Table 5: Results of ablation experiments

Component		TACRED			SemEval 2010		
		P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
	Our model	73.7	70.0	71.8	89.2	93.1	91.1
w/o	Attention residual connection	73.8	69.1	71.4	89.5	91.9	90.6
w/o	AAM	72.0	70.0	71.0	89.2	90.9	90.1
w/o	Attention residual connection, AAM	70.8	70.9	70.8	88.4	90.3	89.4

According to the results shown in the ablation experiment table, (1) when the entity attention residual module is removed, the model's performance decreases by 0.4% on the TACRED dataset and by 0.5% on the SemEval 2010 dataset. The entity attention residual module captures and processes information that sentences focus on entities and mitigates the impact of gradient vanishing problems through cross-layer connections. The information focused on by entities is crucial for understanding the semantics of sentences. Removing this module may lead to the model's inability to fully utilize entity information, thereby affecting overall performance. (2) After removing the adjacency matrix, the model's performance decreased by 0.8% on the TACRED dataset and by 1% on the SemEval

2010 dataset. Our model's adaptive adjacency matrix is initialized by calculating the association degree between tokens in a sentence and participates in model training. The stronger the semantic dependency between tokens in a sentence, the greater the weight at the corresponding position in the adaptive matrix. Removing the adaptive module means the model loses the ability to utilize these semantic dependencies, leading to difficulties in encoding the semantic structure of sentences. (3) When both the adjacency matrix and the entity attention residual module were removed, the F1 scores on the TACRED and SemEval 2010 datasets were 70.8% and 89.4%, respectively. This decline indicates that each module significantly impacts the overall performance. Our model demonstrates that by dynamically learning the adjacency matrix through graph autoencoding, it is practical to automatically capture the semantic relationships between tokens for encoding sentence-level structural information, even without adding any external prior knowledge.

4.5 Performance of the Adjacency Matrix

To demonstrate the effectiveness of the dynamically learned adjacency matrix, we compared it with a randomly initialized fixed adjacency matrix. In this experiment, the values of elements in both matrices were randomly initialized within the range [0, 1]. We evaluated the performance on the TACRED and SemEval 2010 datasets.

The adaptive matrix, initialized using token representations through dot product, offers several advantages: the weights derived from token representations can directly reflect the relevance between tokens, and these weights are further adjusted and optimized during the model training process to meet the needs of specific tasks. This approach contrasts with simple random initialization, where each element of the matrix is assigned a value within the range [0, 1], allowing all tokens in a sentence to be connected but without considering the actual semantic relationships between tokens.

As shown in Fig. 2, on the TACRED dataset, the F1 score of our model using the adaptive adjacency matrix is 0.5% higher than that of the model with a randomly initialized matrix. It also improves performance by approximately 0.5% F1 score on the SemEval 2010 dataset. These experiments clearly demonstrate that the matrix initialized by correlation degree and optimized through dynamic graph learning can more effectively capture the underlying semantic structure between tokens in sentences compared to the randomly initialized adjacency matrix. Consequently, it achieves better prediction results in the relation extraction task.

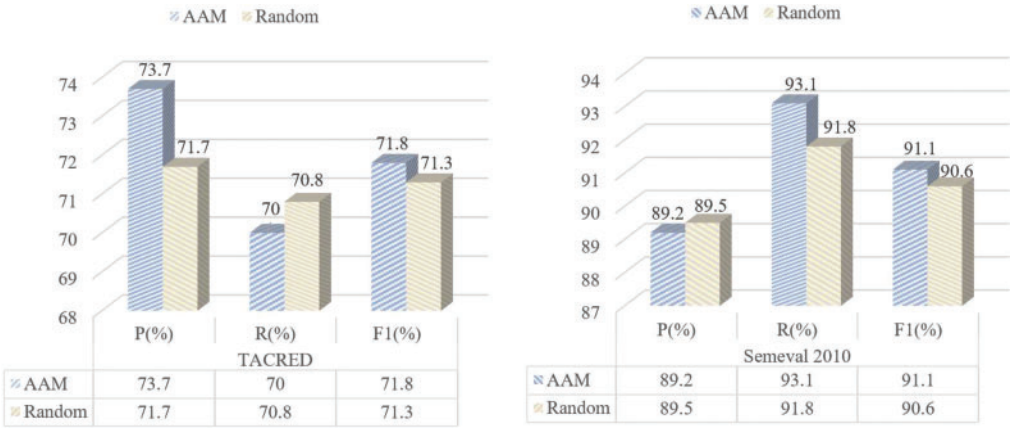

Figure 2: Figures above show the comparison of the AAM and a Random matrix

4.6 Influence of the GCN Layer

As shown in Eq. (3), our model can be stacked with several GCN layers, where the number of layers l can range from 0 to L. To assess the influence of GCN layers, we conducted experiments to analyze the impact of different GCN layer configurations on performance.

As depicted in Fig. 3, we gradually increased the number of GCN layers in our model from 1 to 4. In a single-layer GCN, the network can only learn first-order semantic dependencies between nodes, resulting in weaker performance. On the TACRED dataset, the highest F1 score achieved was 71.8% with two GCN layers. By increasing the number of GCN layers, the network can capture more complex semantic relations, thereby improving model performance. However, as the number of GCN layers continued to increase, performance began to decline. This phenomenon was further verified on the SemEval 2010 dataset, where our model performed best with three GCN layers, achieving an F1 score of 91.1%.

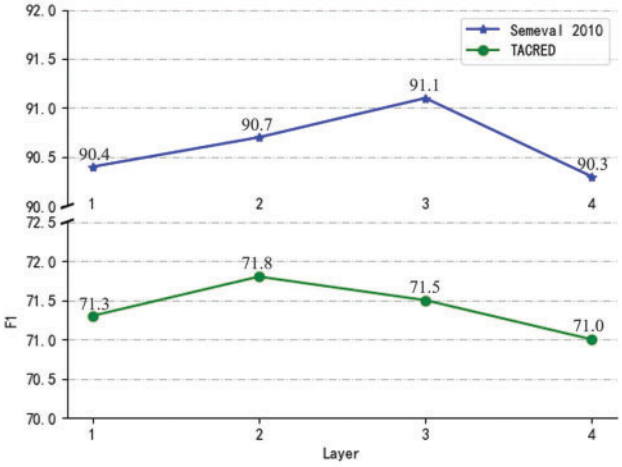

Figure 3: Influence of the number of GCN layers

Although increasing the number of GCN layers can enhance the model's ability to learn higher-order abstract contextual features and semantic dependencies of sentences, it also introduces the common problem of vanishing gradients in deep networks. Vanishing gradients refer to the phenomenon where gradients diminish as the error backpropagates, causing the weights in the deeper layers of the network to update very slowly, thus affecting the overall learning efficiency and performance of the model. Therefore, while there are certain advantages to increasing the number of GCN layers, it is also necessary to weigh the issues that come with deep network structures, such as vanishing gradients.

4.7 Case Study

To better demonstrate the effectiveness of our adaptive adjacency matrix, we conducted a case study comparing our model with a traditional GCN relation extraction model. We selected a sentence from the SemEval 2010 dataset:

"Skype, a free software, allows a <e1>hookup</e1> of multiple computer <e2>users</e2> to join in an online conference call without incurring any telephone costs."

This sentence was fed into our model as well as another model with a randomly initialized adjacency matrix. The results are shown in Fig. 4.

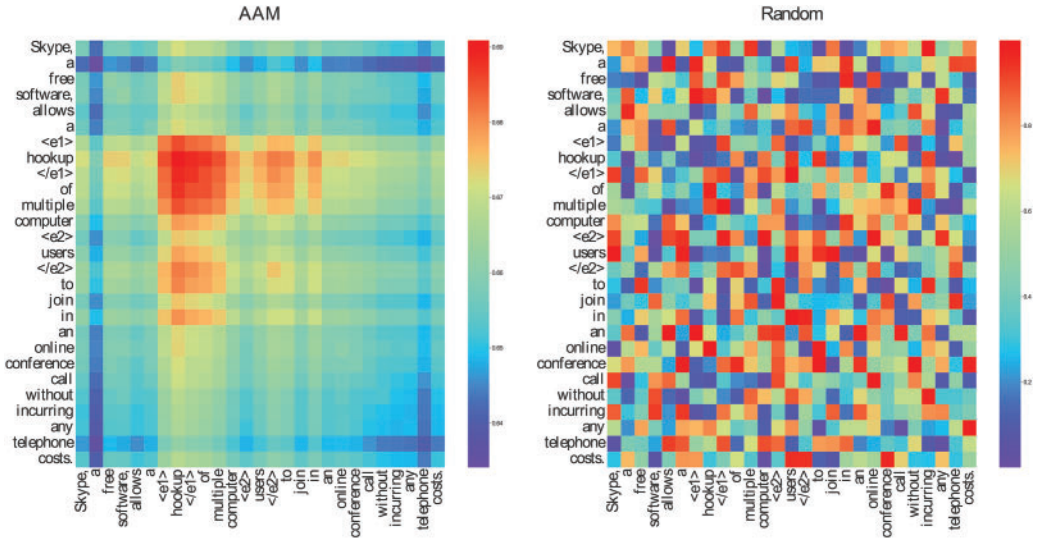

Figure 4: The visualization of AAM and random initialization matrix

In Fig. 4, AAM represents our adaptive adjacency matrix, while Random signifies a randomly initialized matrix. The colors in the figure indicate different intensities of semantic dependence between tokens: darker red signifies a stronger semantic dependency, and darker blue indicates a weaker semantic dependency. Our adaptive adjacency matrix assigns higher weights to tokens that are semantically dependent on the subject "hookup" and the object "users." In contrast, the randomly initialized adjacency matrix appears noisy and fails to reveal any clear semantic patterns between tokens.

4.8 Visualization of the Adaptive Adjacency Matrix

To visualize the ability of adaptive adjacency matrix to learn semantic dependencies in a sentence, we collected three instances from the SemEval 2010 dataset:

(a) "the <e1> ear </e1>of the African <e2>elephant </e2>is significantly larger-measuring 183 cm by 114 cm in the bush elephant."

(b) "I spant a year working for a <e1> softwear </e1> <e2>company</e2> to pay off my college loans".'

(c) "As I was contemplating if I had a favorite in the pink cateory, a <e1> gift </e1>arrived from a sweet <e2>friend</e2>."

In these examples, the named entities are marked by specific token pairs <ei> and </ei> (i $\in$ {1, 2}) to indicate the start and end boundaries of each entity, respectively. The main difference between these sentences is the location of the entity pairs: the front part, the middle part, and the end part of the sentence. These variations are sufficient to compare the semantic dependency patterns between them. These sentences are fed into our optimized GCN, and the values of the generated adjacency matrix are visualized. The results are shown in Fig. 5.

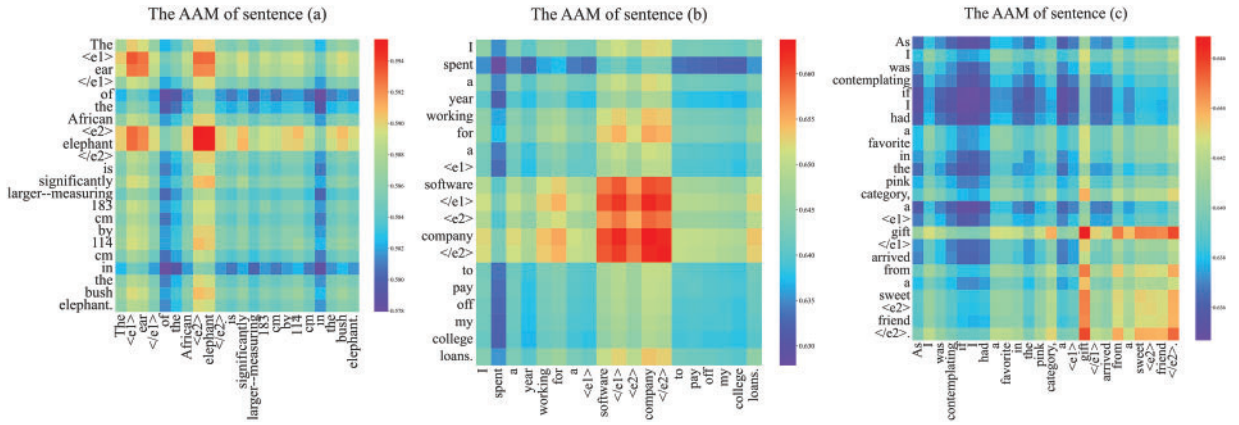

Figure 5: A visualization of AAM with different sentences

In Fig. 5, the color indicates the intensity of semantic dependencies between tokens, with deep red signifying stronger dependencies. The visualized adjacency matrices reveal that the subject "ear" and the object "elephant" in (a), the subject "software" and the object "company" in (b), and the subject "gift" and the object "friend" in (c) are assigned very high weights, indicating a strong correlation between the entity pairs.

In relation extraction, entity markers are highly valuable for supporting the extraction task [48,49]. They enable the deep network to perceive entity boundaries and construct the semantic dependencies between entity boundaries and contextual features. The visualization demonstrates that the adaptive adjacency matrix effectively captures these semantic dependencies indicated by entity markers. The results further validate the capability of our adaptive matrix for relation extraction.

4.9 Error Analysis

Although our model has achieved good results in the relation extraction task, there are still some errors during the training process. By understanding the reasons behind these errors, we can further improve the model. Table 6 presents two error cases identified during the evaluation of the SemEval 2010 dataset. In Table 6, the subject is marked in blue, and the object is marked in green.

Table 6: Error instances

Case	Sentence	True relationship label	Predicted relationship label
1	The subject of "imply" is the source of an implication while the subject of "infer" is the recipient of an implication.	Cause-Effect (e1, e2)	**Entity-Origin (e2, e1)**
2	The room was filled with huge Jack-the-Dripper canvases.	Other	**Content-Container (e2, e1)**

In Case 1, our model predicted the relationship between the entity "subject" and the entity "implication" as Entity-Origin (e2, e1), but the actual relationship type is Cause-Effect (e1, e2). This discrepancy may be due to the presence of two similarly structured clauses in the sentence, leading to a complex sentence structure that could cause model confusion. The model may not have accurately parsed the structure of each clause and failed to associate the verbs (imply and infer) with their respective subjects and objects. To address this issue, we can introduce relative position information in future steps to enhance the model's ability to handle complex sentence structures.

In Case 2, our model incorrectly identified the relationship between "room" and "canvases" as Content-Container (e2, e1), mistakenly implying that the "canvases" are the content of the "room," as if the canvases are stored in the room. However, the sentence does not explicitly express this container-content relationship, so the correct label should be negative (Other). In this example, due to the short length of the sentence and the sparse features, the word "filled" might have led the model to incorrectly assume that the canvases filled the room. To address this issue, our next step could involve extracting multi-granular features to capture semantic and structural information at different levels, thereby enhancing the model's understanding of the text content and overall performance.

5 Conclusion

In this paper, we propose a deep architecture for learning semantic dependencies between tokens in a sentence. Instead of using fixed adjacency matrices initialized by dependency trees or prior knowledge, our graph module adopts an adaptive adjacency matrix to encode semantic dependencies. This matrix is initialized by mutually multiplying the token representations and is subsequently updated during the training process without relying on any external toolkits or prior knowledge. Our method was evaluated on two published datasets, and the results demonstrate that the adaptive adjacency matrix is effective in learning semantic dependencies for relation extraction. Additionally, this model has the potential to be developed for other NLP tasks.

Acknowledgement: None.

Funding Statement: This work was supported by the Technology Projects of Guizhou Province under Grant [2024]003 and National Natural Science Foundation of China (Grant Nos. 62166007, 62066008, 62066007) and Guizhou Provincial Science and Technology Projects under Grant No. ZK[2023]300.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Run Yang, Yanping Chen; data collection: Jiaxin Yan; analysis and interpretation of results: Run Yang, Yanping Chen, Yongbin Qin; draft manuscript preparation: Run Yang, Yanping Chen, Jiaxin Yan, Yongbin Qin. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel and G. Bouchard, "Knowledge graph completion via complex tensor factorization," *J. Mach. Learn. Res.*, vol. 18, no. 130, pp. 1–38, 2017.

[2] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan and D. Zhao, "Relation-aware entity alignment for heterogeneous knowledge graphs," arXiv preprint arXiv:1908.08210, 2019.

[3] C. Quirk and H. Poon, "Distant supervision for relation extraction beyond the sentence boundary," arXiv preprint arXiv:1609.04873, 2016.

[4] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W. Yih, "Cross-sentence N-ary relation extraction with graph LSTMs," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 101–115, 2017. doi: 10.1162/tacl_a_00049.

[5] M. Yang, N. Duan, M. Zhou, and H. C. Rim, "Joint relational embeddings for knowledge based question answering," in *Proc. 2014 Conf. Empirical Methods in Nat. Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 645–650.

[6] M. Yu, W. Yin, K. S. Hasan, C. Santos, B. Xiang and B. Zhou, "Improved neural relation detection for knowledge base question answering," arXiv preprint arXiv:1704.06194, 2017.

[7] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piece wise convolutional neural networks," in *Proc. 2015 Conf. Empir. Methods Natural Lang. Process.*, 2015, pp. 1753–1762.

[8] Y. Chen, K. Wang, W. Yang, Y. Qing, R. Huang and P. Chen, "A multi-channel deep neural network for relation extraction," *IEEE Access*, vol. 8, pp. 13195–13203, 2020. doi: 10.1109/ACCESS.2020.2966303.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[10] Y. Zhang, P. Qi, and C. D. Manning, "Graph convolution over pruned dependency trees improves relation extraction," arXiv preprint arXiv:1809.10185, 2018.

[11] Q. Zhao, F. Yang, D. An, and J. Lian, "Modeling structured dependency tree with graph convolutional networks for aspect-level sentiment classification," *Sensors*, vol. 24, no. 2, 2024, Art. no. 418. doi: 10.3390/s24020418.

[12] Y. Chen, Q. Zheng, and W. Zhang, "Omni word feature and soft constraint for Chinese relation extraction," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguist.*, 2014, pp. 572–581.

[13] Y. Chen, W. Yang, K. Wang, Y. Qin, R. Huang and Q. Zheng, "A neuralized feature engineering method for entity relation extraction," *Neural Netw.*, vol. 141, no. 1, pp. 249–260, 2021. doi: 10.1016/j.neunet.2021.04.010.

[14] M. Miwa and M. Bansal, "End-to-end relation extraction using LSTMs on sequences and tree structures," arXiv preprint arXiv:1601.00770, 2016.

[15] J. Lee, S. Seo, and Y. S. Choi, "Semantic relation classification via bidirectional LSTM networks with entity-aware attention using latent entity typing," *Symmetry*, vol. 11, no. 6, 2019, Art. no. 785. doi: 10.3390/sym11060785.

[16] Z. Guo, Y. Zhang, and W. Lu, "Attention guided graph convolutional networks for relation extraction," arXiv preprint arXiv:1906.07510, 2019.

[17] L. Zhou, T. Wang, H. Qu, L. Huang, and Y. Liu, "A weighted GCN with logical adjacency matrix for relation extraction," in *ECAI 2020*. Amsterdam, IOS Press, 2020, pp. 2314–2321.

[18] Y. Tian, G. Chen, Y. Song, and X. Wan, "Dependency-driven relation extraction with attentive graph convolutional networks," in *Proc. 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Joint Conf. Nat. Lang. Process.*, 2021, pp. 4458–4471.

[19] M. Zhang and T. Qian, "Multi-scale feature and metric learning for relation extraction," arXiv preprint arXiv:2107.13425, 2021.

[20] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, "Composition-based multi relational graph convolutional networks," arXiv preprint arXiv:1911.03082, 2019.

[21] J. Xu, Y. Chen, Y. Qin, R. Huang, and Q. Zheng, "A feature combination based graph convolutional neural network model for relation extraction," *Symmetry*, vol. 13, no. 8, 2021, Art. no. 1458. doi: 10.3390/sym13081458.

[22] N. Kambhatla, "Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction," in *Proc. ACL Interact. Poster and Demonstration Sessions*, Ann Arbor, Michigan, 2004, pp. 178–181.
[23] G. Zhou, J. Su, J. Zhang, and M. Zhang, "Exploring various knowledge in relation extraction," in *Proc. 43rd Annu. Meet. Assoc. Comput. Linguist. (ACL'05)*, Ann Arbor, Michigan, 2005, pp. 427–434.
[24] Y. Li *et al.*, "Relation extraction in biomedical texts based on multi-head attention model with syntactic dependency feature: Modeling study," *JMIR Med. Inform.*, vol. 10, no. 10, 2022, Art. no. e41136. doi: 10.2196/41136.
[25] J. Achiam *et al.*, "GPT-4 technical report," arXiv preprint arXiv: 2303.08774, 2023.
[26] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
[27] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguist.*, vol. 8, pp. 64–77, 2020. doi: 10.1162/tacl_a_00300.
[28] H. Touvron *et al.*, "LlaMa: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.
[29] H. Fei *et al.*, "LasUIE: Unifying information extraction with latent adaptive structure-aware generative language model," *Adv. Neural Inf. Process Syst.*, vol. 35, pp. 15460–15475, 2022.
[30] J. Li *et al.*, "Unified named entity recognition as word-word relation classification," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 10965–10973.
[31] C. Cho and Y. S. Choi, "Dependency tree positional encoding method for relation extraction," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, New York, NY, USA, 2021, pp. 1012–1020.
[32] A. P. B. Veyseh, F. Dernoncourt, D. Dou, and T. H. Nguyen, "Exploiting the syntax-model consistency for neural relation extraction," in *Proc. 58th Annu. Meet. Assoc. Comput. Linguist.*, 2020, pp. 8021–8032.
[33] X. Liang, S. Wu, M. Li, and Z. Li, "Modeling multi-granularity hierarchical features for relation extraction," arXiv preprint arXiv:2204.04437, 2022.
[34] C. Wei, Y. Chen, K. Wang, Y. Qin, R. Huang and Q. Zheng, "APRE: Annotation-aware prompt-tuning for relation extraction," *Neural Process. Lett.*, vol. 56, no. 2, 2024, Art. no. 62. doi: 10.1007/s11063-024-11437-y.
[35] H. Fei, J. Li, S. Wu, C. Li, D. Ji and F. Li, "Global inference with explicit syntactic and discourse structures for dialogue-level relation extraction," in *Proc. IJCAI*, Vienna, Austria, 2022, pp. 4107–4113.
[36] Y. Dong and X. Xu, "Weighted-dependency with attention-based graph convolutional network for relation extraction," *Neural Process. Lett.*, vol. 55, no. 9, pp. 12121–12142, 2023. doi: 10.1007/s11063-023-11412-z.
[37] W. Ai, Y. Shou, T. Meng, and K. Li, "DER-GCN: Dialog and event relation-aware graph convolutional neural network for multimodal dialog emotion recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, 2024. doi: 10.1109/TNNLS.2024.3367940.
[38] M. Zuo, Z. Song, Y. Liu, Q. Zhang, Y. Cai and D. Wu, "Event type and relationship extraction based on dependent syntactic semantic augmented graph networks," 2024. doi: 10.21203/rs.3.rs-3923678/v1.
[39] S. Cui *et al.*, "Enhancing multimodal entity and relation extraction with variational information bottleneck," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 32, pp. 1274–1285, 2024. doi: 10.1109/TASLP.2023.3345146.
[40] C. Sun *et al.*, "Joint type inference on entities and relations via graph convolutional networks," in *Proc. 57th Annu. Meet. Assoc. Comput. Linguist.*, Florence, Italy, 2019, pp. 1361–1370.
[41] H. Zhu, Y. Lin, Z. Liu, J. Fu, T. Chua and M. Sun, "Graph neural networks with generated parameters for relation extraction," arXiv preprint arXiv:1902.00756, 2019.
[42] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Conf. Empir. Methods Nat. Lang. Process.*, Copenhagen, Denmark, 2017, pp. 35–45.
[43] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng and Z. Jin, "Classifying relations via long short term memory networks along shortest dependency paths," in *Proc. 2015 Conf. Empir. Methods Nat. Lang. Process.*, Lisbon, Portugal, 2015, pp. 1785–1794.

[44] Y. Hu, H. Shen, W. Liu, F. Min, X. Qiao and K. Jin, "A graph convolutional network with multiple dependency representations for relation extraction," *IEEE Access*, vol. 9, pp. 81575–81587, 2021. doi: 10.1109/ACCESS.2021.3086480.

[45] D. Zhang, Z. Liu, W. Jia, F. Wu, H. Liu and J. Tan, "Dual attention graph convolutional network for relation extraction," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 2, pp. 1–14, 2023. doi: 10.1109/TKDE.2023.3289879.

[46] J. Liao, Y. Du, J. Hu, H. Li, X. Li and X. Chen, "A contextual dependency-aware graph convolutional network for extracting entity relations," *Expert. Syst. Appl.*, vol. 239, no. 10, 2024, Art. no. 122366. doi: 10.1016/j.eswa.2023.122366.

[47] S. Wu and Y. He, "Enriching pre-trained language model with entity information for relation clas sification," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manag.*, Boise, USA, 2019, pp. 2361–2364.

[48] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski, "Matching the blanks: Distributional similarity for relation learning," arXiv preprint arXiv:1906.03158, 2019.

[49] Y. Qin *et al.*, "Entity relation extraction based on entity indicators," *Symmetry*, vol. 13, no. 4, 2021, Art. no. 539. doi: 10.3390/sym13040539.